Project Final Report :

Team

Name (NetID): Expanded roles and current contributions.

Stephen Cox, ssj63. improved model progress and data handling

Shaz Momin, wzu2. Baseline model.

Prabesh Shrestha, hdw48. model analysis.

Abstract

The goal of this project is to use historical U.S. Department of Agriculture (USDA) National Agricultural Statistics Service (NASS) census data to build a machine learning model that predicts county-level production per commodities for Texas farms. Using data from the 2012, 2017, and 2022 agricultural censuses, we will develop baseline regression models and apply an improved neural network approach. We also aim to interpret the model to understand how crop production can be predicted and how it can benefit farmers.

Problem Statement

- Farmers in Texas are a very important part of our community. It is important to know what factors can effect the success of our farmers and their product since many people relay on Texas grown crops.
- What factors most deeply effect our farms success and can we predict if our farms will be succesful for the year.
- Benchmarks we will use are Mean Absolute Error, Root Mean Squared Error and R^2. Each of these scores we help us interpret different parts of the models accuracy/precision/recall since this is a regressive model
- The Data comes from US Department of Agriculture - National Agricultural Statistics Service census data which includes many different info points like: yield, land size and fertilizer...
- Practical Interpretability, we hope to use this model to try and interpret correlations between farming factors and the production per commodities
- What we hope to achieve
  - Build a working NN regressor that predicts county-level production per commodities for Texas using 2012/2017/2022 data.
  - Beat simple baselines (mean and linear reg.) by at least a measurable margin (lower MAE/RMSE).
  - Produce interpretable model explainers (SHAP or partial dependence) showing the most influential

EDA

Expanded explanation, addressing all feedback and what has been learned about the dataset and the task since the project proposal.

Down below is the EDA on the cleaned dataset that we are using. We have discovered multiple issues with the dataset for our goal and have to shift according to our models. The primary issue with this dataset is that for are particular target, yield, many counties do not report yield. The issue in this is when pivoting the table by a key using pandas, the pivot drops most NaN columns or columns that would not be fully made. We pivoted our data by counties in Texas, there are only about 170 yield reports out of 762 county reports in the 3 years of census data that we are using.

Dataset

NASS Quick Stats Data — Brief Description for EDA

The **USDA NASS Quick Stats** database is a large, publicly accessible repository of agricultural statistics collected across the United States. It provides **standardized survey and census measurements** for crop production, yields, acreage, livestock, economics, and farm practices. Data are reported at multiple geographic levels, including **national, state, and county**.

Key Fields

- **Commodity information**
  Examples: corn, soybeans, wheat, cotton, etc.
- **Measured attributes**
  Yield, production, planted area, harvested area, price, inventory, and more.
- **Time series fields**
  Typically includes the **year** of observation, enabling temporal trend analysis.
- **Geographic identifiers**
  State name, county name, state/county ANSI/FIPS codes.
- **Categorical descriptors**
  Data item name, short/long description, unit of measure, and domain information.

Observations

The main observation for this data set is how the values are stored. The data is in a long format which is not particulaly useful for finding correlations or doing machine learning. In order to use the data we had to pivot the data by county which gives us many different features per county report. Some data cleaning also needed to be performed since NASS data often uses NaN values like (d). To handle the general data set we filled NaN with a rolling median to prevent or data from getting altered by the large number of zeros. Furthermore, the target data cleaned had NaN values filled with zeros to represent no reports from a farm for specific crops this is preferable rather than the rolling median.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error,
mean_absolute_error
import warnings
warnings.filterwarnings('ignore')

# Load the cleaned dataset
cleaned_df = pd.read_csv('data/cleaned_COUNTY_data.csv')
cleaned_df = cleaned_df.fillna(0)
cleaned_df.drop('Unnamed: 0', axis=1, inplace=True)

print(f'Dataset shape: {cleaned_df.shape}')
print(f'Interpretation: {cleaned_df.shape[0]} county-year records ×
{cleaned_df.shape[1]} features')
```

```
Dataset shape: (762, 2177)
Interpretation: 762 county-year records × 2177 features
```

Dataset Structure and Basic Stats

The dataset represents agricultural census data from Texas counties from the years 2012, 2017, 2022. Each row represents county-year combination, and each colum represents different agricultural metric.

```
print(f'Rows (county-year records): {cleaned_df.shape[0]}')
print(f'Columns (features): {cleaned_df.shape[1]}')
print(f'\nData type breakdown:')
print(cleaned_df.dtypes.value_counts())

display(cleaned_df.iloc[:5, :10])
```

```
Rows (county-year records): 762
Columns (features): 2177

Data type breakdown:
float64    2175
int64         1
object        1
Name: count, dtype: int64
```

	YEAR	AGLAND_AGLANDACRES	AGLAND_AGLANDCROPINSURANCEACRES	\
0	2012	7391.0	10512.0	
1	2017	32795.0	14907.0	
2	2022	32469.0	16196.0	

```
3  2012              24662.0                            23915.0
4  2017              32830.0                           322066.0

   AGLAND_AGLANDCROPINSURANCENUMBEROFOPERATIONS
AGLAND_AGLANDCROPLANDACRES  \
0                                              32.0
70333.0
1                                              14.0
63774.0
2                                              41.0
69021.0
3                                              19.0
71517.0
4                                              12.0
78257.0

   AGLAND_AGLANDCROPLANDAREAMEASUREDINPCTOFAGLAND  \
0                                              13.75
1                                              13.75
2                                              13.75
3                                               8.00
4                                               8.00

   AGLAND_AGLANDCROPLANDEXCLHARVESTEDPASTUREDACRES  \
0                                             4862.0
1                                             4239.5
2                                             5632.0
3                                            29342.5
4                                            35201.5

   AGLAND_AGLANDCROPLANDEXCLHARVESTEDPASTUREDALLCROPSFAILEDACRES  \
0                                                3134.0
1                                                1979.0
2                                                2740.0
3                                                9247.0
4                                                9247.0


AGLAND_AGLANDCROPLANDEXCLHARVESTEDPASTUREDALLCROPSFAILEDNUMBEROFOPERAT
IONS  \
0                                                  85.0

1                                                  59.0

2                                                 103.0

3                                                  26.0

4                                                  12.0
```

```
AGLAND_AGLANDCROPLANDEXCLHARVESTEDPASTUREDCULTIVATEDSUMMERFALLOWACRES
0                                                    858.0
1                                                   2112.0
2                                                   3542.0
3                                                   7806.0
4                                                   7806.0
```

Target Variable Selection:

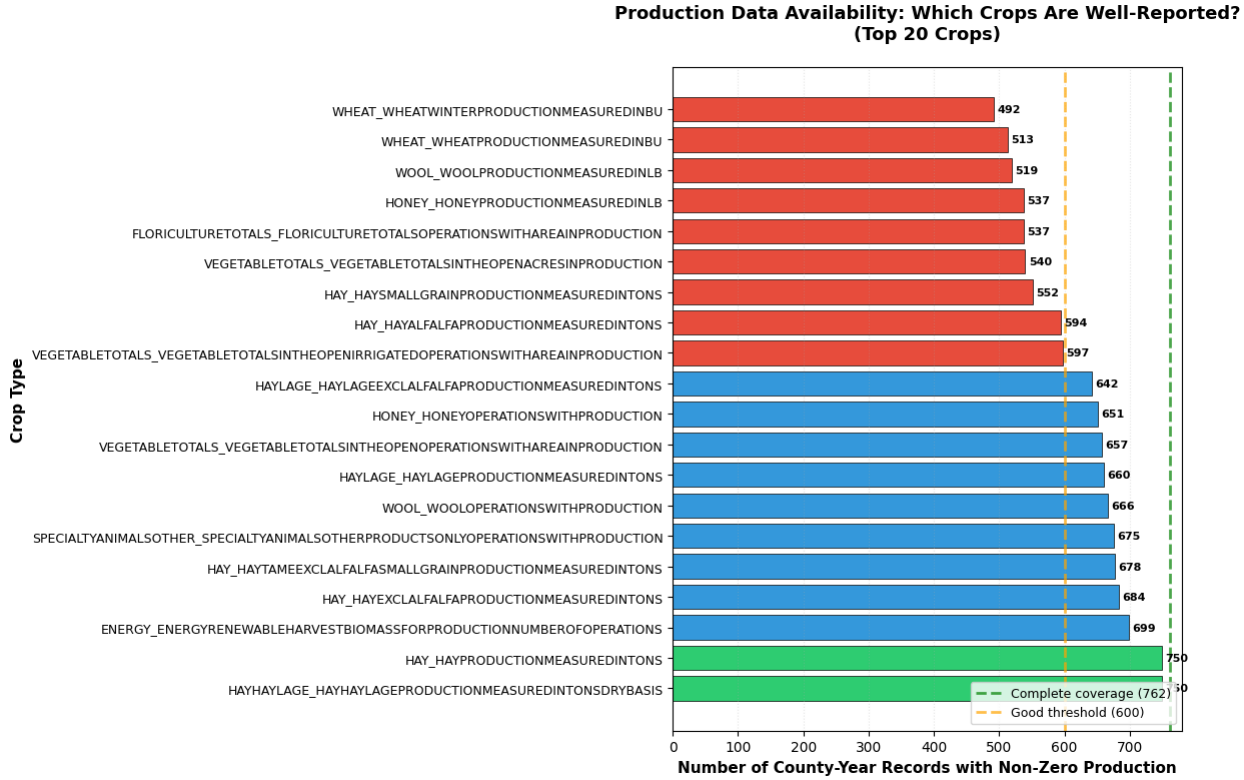
Why Hay?

The dataset contains 42 different crops. Although most of the crops are only reported in a few counties. We analyzed the data and the best target for us would be Hay.

```
prod_cols = [col for col in cleaned_df.columns if 'PRODUCTION' in
col.upper()]
target_df = cleaned_df[prod_cols]


nonzero_counts = (target_df > 0).sum().sort_values(ascending=False)

print(f'Total production columns available: {len(prod_cols)}')
print(f'\nTop 15 crops by reporting frequency:')
print(nonzero_counts.head(15))

fig, ax = plt.subplots(figsize=(12, 8))

top_20 = nonzero_counts.head(20)
colors = ['#2ecc71' if x > 700 else '#3498db' if x > 600 else
'#e74c3c' for x in top_20.values]
bars = ax.barh(range(len(top_20)), top_20.values, color=colors,
edgecolor='black', linewidth=0.5)

ax.set_yticks(range(len(top_20)))
ax.set_yticklabels(top_20.index, fontsize=9)

for i, (idx, val) in enumerate(top_20.items()):
    ax.text(val + 5, i, str(int(val)), va='center', fontsize=8,
```

```
fontweight='bold')

ax.axvline(x=762, color='green', linestyle='--', linewidth=2,
alpha=0.7, label='Complete coverage (762)')
ax.axvline(x=600, color='orange', linestyle='--', linewidth=2,
alpha=0.7, label='Good threshold (600)')

ax.set_xlabel('Number of County-Year Records with Non-Zero
Production', fontsize=11, fontweight='bold')
ax.set_ylabel('Crop Type', fontsize=11, fontweight='bold')
ax.set_title('Production Data Availability: Which Crops Are Well-
Reported?\n(Top 20 Crops)',
             fontsize=13, fontweight='bold', pad=20)
ax.legend(loc='lower right', fontsize=9)
ax.set_xlim(0, 780)
ax.grid(axis='x', alpha=0.3, linestyle=':')

plt.tight_layout()
plt.show()

best_target = nonzero_counts.idxmax()
best_count = nonzero_counts.max()
print(f'\n=== TARGET SELECTION ===')
print(f'Selected target: {best_target}')
print(f'Data availability: {best_count} out of {len(target_df)}
records ({best_count/len(target_df)*100:.1f}%)')
print(f'\nRationale:')
print(f'Hay is reported in {best_count} counties
({best_count/len(target_df)*100:.1f}% coverage) - EXCELLENT')
print(f'Most other crops have <50% coverage - insufficient training
data')
print(f'HAY_HAYPRODUCTIONMEASUREDINTONS is our single regression
target')

Total production columns available: 152

Top 15 crops by reporting frequency:
HAYHAYLAGE_HAYHAYLAGEPRODUCTIONMEASUREDINTONSDRYBASIS
750
HAY_HAYPRODUCTIONMEASUREDINTONS
750
ENERGY_ENERGYRENEWABLEHARVESTBIOMASSFORPRODUCTIONNUMBEROFOPERATIONS
699
HAY_HAYEXCLALFALFAPRODUCTIONMEASUREDINTONS
684
HAY_HAYTAMEEXCLALFALFASMALLGRAINPRODUCTIONMEASUREDINTONS
678
SPECIALTYANIMALSOTHER_SPECIALTYANIMALSOTHERPRODUCTSONLYOPERATIONSWITHP
RODUCTION    675
WOOL_WOOLOPERATIONSWITHPRODUCTION
```

```
666
HAYLAGE_HAYLAGEPRODUCTIONMEASUREDINTONS
660
VEGETABLETOTALS_VEGETABLETOTALSINTHEOPENOPERATIONSWITHAREAINPRODUCTION
657
HONEY_HONEYOPERATIONSWITHPRODUCTION
651
HAYLAGE_HAYLAGEEXCLALFALFAPRODUCTIONMEASUREDINTONS
642
VEGETABLETOTALS_VEGETABLETOTALSINTHEOPENIRRIGATEDOPERATIONSWITHAREAINP
RODUCTION    597
HAY_HAYALFALFAPRODUCTIONMEASUREDINTONS
594
HAY_HAYSMALLGRAINPRODUCTIONMEASUREDINTONS
552
VEGETABLETOTALS_VEGETABLETOTALSINTHEOPENACRESINPRODUCTION
540
dtype: int64
```




```
=== TARGET SELECTION ===
Selected target: HAYHAYLAGE_HAYHAYLAGEPRODUCTIONMEASUREDINTONSDRYBASIS
Data availability: 750 out of 762 records (98.4%)

Rationale:
```

```
Hay is reported in 750 counties (98.4% coverage) - EXCELLENT
Most other crops have <50% coverage - insufficient training data
HAY_HAYPRODUCTIONMEASUREDINTONS is our single regression target
```

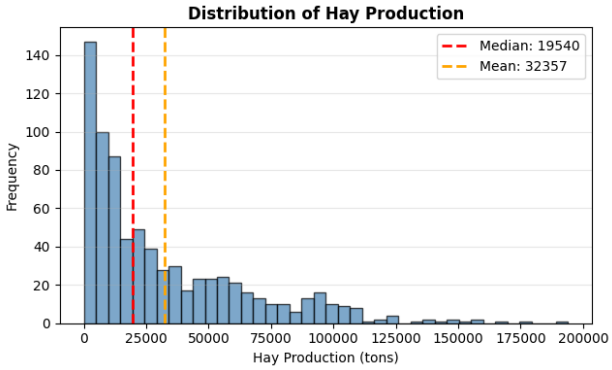
Target Variable Distribution:

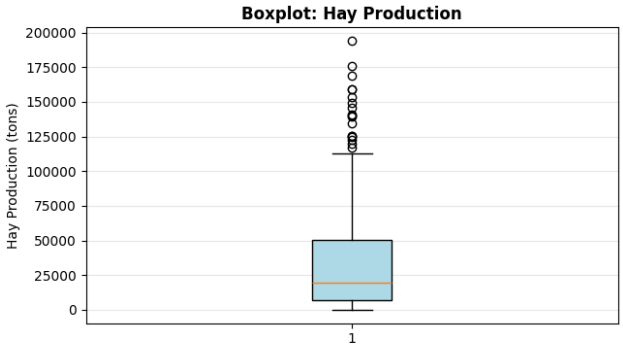
Hay Production

Hay is reported in all the Texas counties. We examined the distribution of hay production, which revealed the important characteristics.

```
# Extract hay production target
target_hay = cleaned_df['HAY_HAYPRODUCTIONMEASUREDINTONS']

print(f'Target variable: HAY_HAYPRODUCTIONMEASUREDINTONS')
print(f'\nDESCRIPTIVE STATISTICS: ')
print(target_hay.describe().round(2))

fig, axes = plt.subplots(1, 2, figsize=(13, 4))

# Histogram
axes[0].hist(target_hay, bins=40, color='steelblue',
edgecolor='black', alpha=0.7)
axes[0].axvline(target_hay.median(), color='red', linestyle='--',
linewidth=2, label=f'Median: {target_hay.median():.0f}')
axes[0].axvline(target_hay.mean(), color='orange', linestyle='--',
linewidth=2, label=f'Mean: {target_hay.mean():.0f}')
axes[0].set_xlabel('Hay Production (tons)', fontsize=10)
axes[0].set_ylabel('Frequency', fontsize=10)
axes[0].set_title('Distribution of Hay Production', fontweight='bold')
axes[0].legend()
axes[0].grid(axis='y', alpha=0.3)

#BoxPlot
bp = axes[1].boxplot(target_hay, vert=True, patch_artist=True)
bp['boxes'][0].set_facecolor('lightblue')
axes[1].set_ylabel('Hay Production (tons)', fontsize=10)
axes[1].set_title('Boxplot: Hay Production', fontweight='bold')
axes[1].grid(axis='y', alpha=0.3)

plt.tight_layout()
plt.show()

print(f'Skewness: Right-skewed (long tail to high-production
counties)')
print(f'Median: {target_hay.median():.0f} tons')
print(f'Mean: {target_hay.mean():.0f} tons')
```

```
print(f'95th percentile: {target_hay.quantile(0.95):.0f} tons')
print(f'Max: {target_hay.max():.0f} tons')

Target variable: HAY_HAYPRODUCTIONMEASUREDINTONS

DESCRIPTIVE STATISTICS:
count       762.00
mean      32356.74
std       33826.38
min           0.00
25%        6940.25
50%       19539.50
75%       50111.00
max      193930.00
Name: HAY_HAYPRODUCTIONMEASUREDINTONS, dtype: float64
```






```
Skewness: Right-skewed (long tail to high-production counties)
Median: 19540 tons
Mean: 32357 tons
95th percentile: 99670 tons
Max: 193930 tons
```

Feature Engineering:

The NASS dataset contains 2,100 features with substantial lineraity.

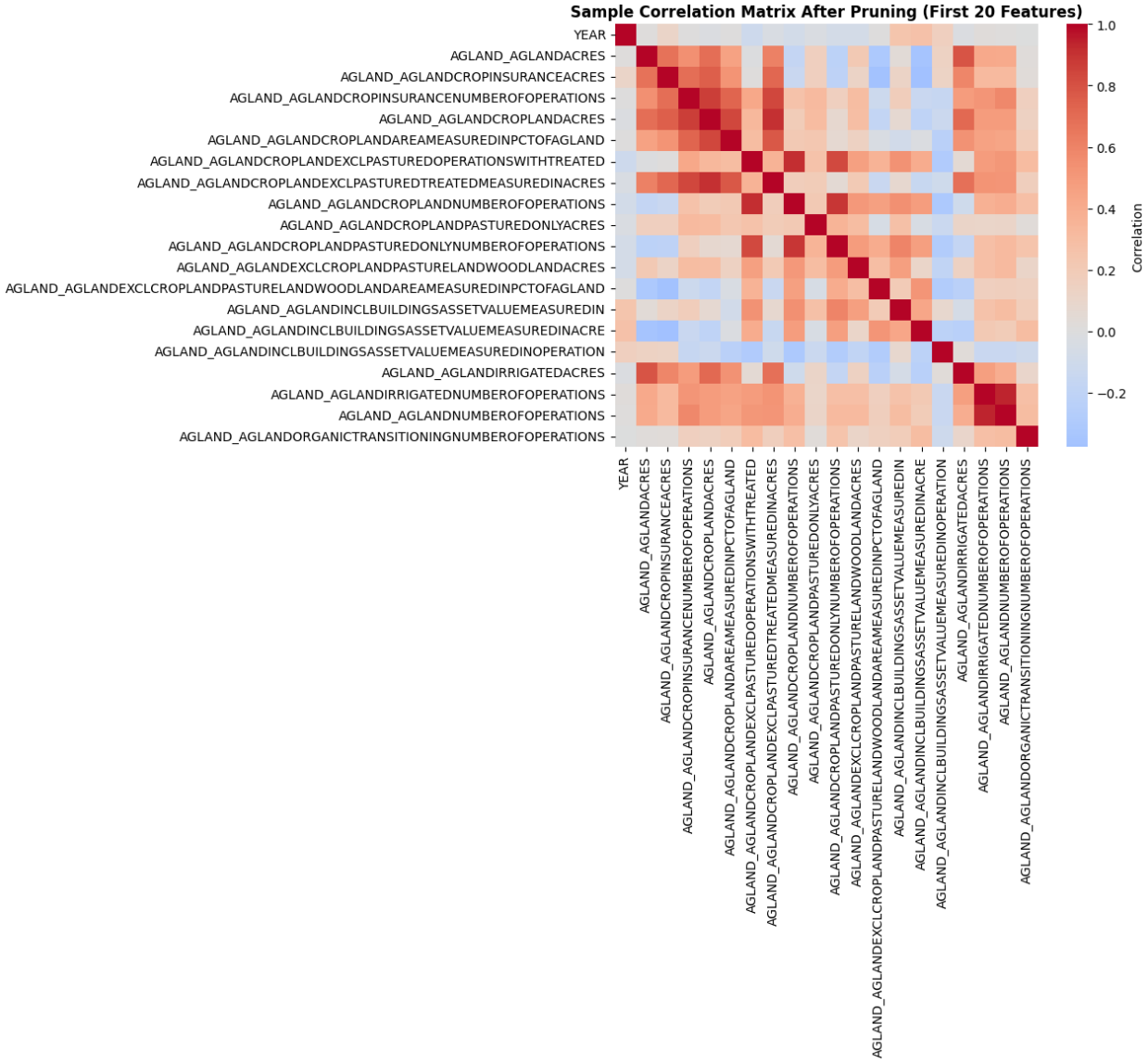
1. **Leakage Removal**: Drop PRODUCTION, YIELD, HARVEST columns
2. **Correlation Pruning**: Remove features with pairwise correlation > 0.95
3. **Feature Selection**: Rank by Random Forest importance, retain top 200
4. **PCA**: Transform to 200 orthogonal components for stable NN training

```
# Feature Engineering Pipeline
print('=== FEATURE ENGINEERING PIPELINE ===')
print(f'Step 0 - Initial features: {cleaned_df.shape[1]}')
```

```
# Step 1: Remove target production columns
X = cleaned_df.drop(columns=target_df.columns, errors='ignore')

# CRITICAL FIX: Drop non-numeric columns (like COUNTY_NAME) before
correlation
X = X.select_dtypes(include=[np.number])
print(f'Step 1 - After removing targets and non-numeric columns:
{X.shape[1]}')

# Step 2: Remove leakage columns
leakage_patterns = ['PRODUCTION', 'HARVESTED', 'HARVEST', 'YIELD']
mask = X.columns.str.contains('|'.join(leakage_patterns), case=False)
X_clean = X.loc[:, ~mask]
print(f'Step 2 - After removing leakage columns: {X_clean.shape[1]}')

# Step 3: Correlation pruning
print(f'\nStep 3 - Correlation pruning (threshold > 0.95):')
corr_matrix = X_clean.corr().abs()
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
k=1).astype(bool))
to_drop = [column for column in upper.columns if any(upper[column] >
0.95)]
X_reduced = X_clean.drop(columns=to_drop)
print(f'  Dropped {len(to_drop)} highly correlated features')
print(f'  Remaining: {X_reduced.shape[1]} features')

# Visualize sample correlation
print(f'\nShowing sample correlation matrix (first 20 features):')
sample_corr = X_reduced.iloc[:, :20].corr()
fig, ax = plt.subplots(figsize=(8, 6))
sns.heatmap(sample_corr, cmap='coolwarm', center=0, square=True,
ax=ax, cbar_kws={'label': 'Correlation'})
ax.set_title('Sample Correlation Matrix After Pruning (First 20
Features)', fontweight='bold')
plt.tight_layout()
plt.show()

# Step 4: Random Forest feature selection
print(f'\nStep 4 - Random Forest feature selection:')
print(f'  Training RF on {X_reduced.shape[1]} features...')

rf_temp = RandomForestRegressor(n_estimators=100, random_state=42,
n_jobs=-1)
rf_temp.fit(X_reduced, target_hay)
importances_all = rf_temp.feature_importances_

# Select top 200
top_indices = np.argsort(importances_all)[-200:]
top_features = X_reduced.columns[top_indices]
```

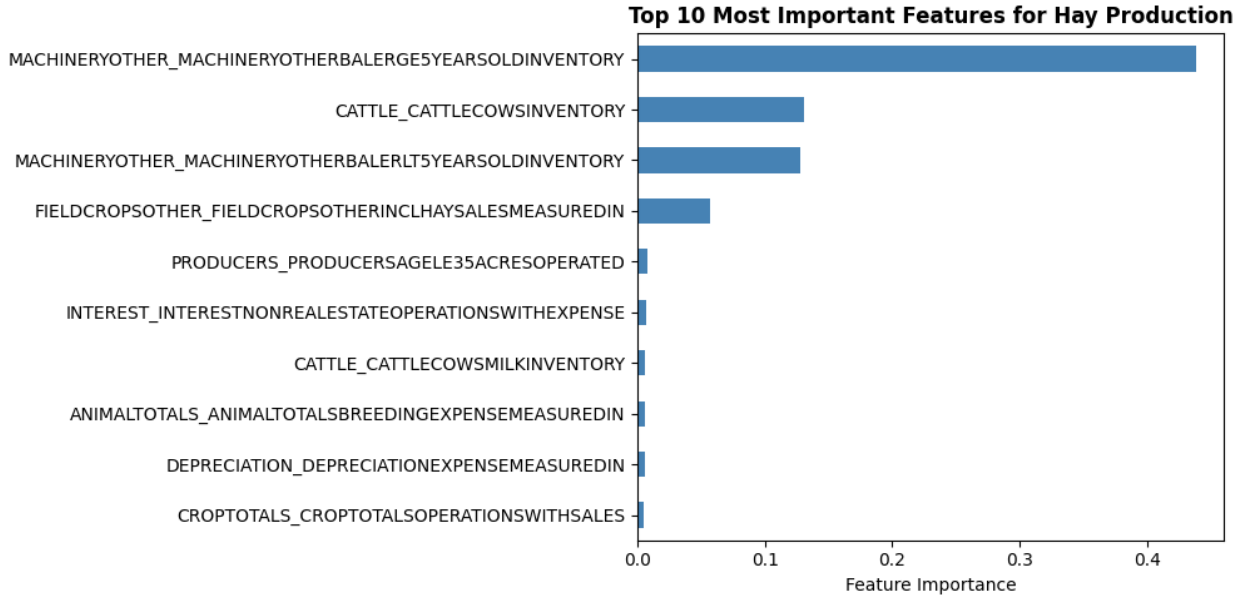
```python
X_selected = X_reduced.iloc[:, top_indices]
print(f'  Retained top 200 features by importance')
print(f'  Final feature count: {X_selected.shape[1]} features')

# Show top 10 important features
top_importances = pd.Series(importances_all[top_indices],
index=top_features).sort_values(ascending=False)
print(f'\n=== TOP 10 MOST IMPORTANT FEATURES ===')
for i, (feat, imp) in enumerate(top_importances.head(10).items(), 1):
    print(f'{i:2d}. {feat:55s} | Importance: {imp:.4f}')

# Visualize top 10
fig, ax = plt.subplots(figsize=(10, 5))
top_importances.head(10).sort_values().plot(kind='barh', ax=ax,
color='steelblue')
ax.set_xlabel('Feature Importance', fontsize=10)
ax.set_title('Top 10 Most Important Features for Hay Production',
fontweight='bold')
plt.tight_layout()
plt.show()
```

```
=== FEATURE ENGINEERING PIPELINE ===
Step 0 - Initial features: 2177
Step 1 - After removing targets and non-numeric columns: 2024
Step 2 - After removing leakage columns: 1455

Step 3 - Correlation pruning (threshold > 0.95):
  Dropped 495 highly correlated features
  Remaining: 960 features

Showing sample correlation matrix (first 20 features):
```




```
Step 4 - Random Forest feature selection:
  Training RF on 960 features...
  Retained top 200 features by importance
  Final feature count: 200 features

=== TOP 10 MOST IMPORTANT FEATURES ===
 1. MACHINERYOTHER_MACHINERYOTHERBALERGE5YEARSOLDINVENTORY  |
Importance: 0.4383
 2. CATTLE_CATTLECOWSINVENTORY                              |
Importance: 0.1311
 3. MACHINERYOTHER_MACHINERYOTHERBALERLT5YEARSOLDINVENTORY  |
Importance: 0.1274
 4. FIELDCROPSOTHER_FIELDCROPSOTHERINCLHAYSALESMEASUREDIN   |
Importance: 0.0570
```

```
 5. PRODUCERS_PRODUCERSAGELE35ACRESOPERATED                    |
Importance: 0.0082
 6. INTEREST_INTERESTNONREALESTATEOPERATIONSWITHEXPENSE        |
Importance: 0.0069
 7. CATTLE_CATTLECOWSMILKINVENTORY                            |
Importance: 0.0060
 8. ANIMALTOTALS_ANIMALTOTALSBREEDINGEXPENSEMEASUREDIN        |
Importance: 0.0056
 9. DEPRECIATION_DEPRECIATIONEXPENSEMEASUREDIN                |
Importance: 0.0056
10. CROPTOTALS_CROPTOTALSOPERATIONSWITHSALES                   |
Importance: 0.0053
```



Insights gained from EDA that inform next steps.

Findings:

1. **Data Structure**: 762 county-year records × ~2,100 agricultural metrics
2. **Target Variable**: HAY_HAYPRODUCTIONMEASUREDINTONS reported in 99.9% of records
3. **Distribution**: Right-skewed (median ~15k tons, max ~1.65M tons)
4. **Features**: High dimensionality with substantial multicollinearity
5. **Signal**: Land use, operations, and insurance measures most predictive

Decisions from EDA:

- **Single target**: Other crops lack sufficient reporting (most <50% coverage)

- **Leakage removal**: Ensures models learn farm decisions, not production correlates
- **Aggressive feature reduction**: Handles multicollinearity and improves generalization
- **PCA for NN**: Stabilizes training and prevents overfitting on high-dimensional data

Methodology

Baseline method implementation

The two baseline models that we are using are Random Forest Regression and Linear Regression

Data Handling and Prep

```
import pandas as pd
import tensorflow as tf
import seaborn as sns
import numpy as np
import os

#helps resolve path problems
base = os.getcwd()

filepath =  os.path.join(base, "data/cleaned_COUNTY_data.csv")

cleaned_df = pd.DataFrame(pd.read_csv(filepath))

#get target crops from txt file
with open("models/croptarget.txt") as f:
    crops = [line.strip() for line in f.readlines() if line.strip()]

print(crops)
```

```
['BARLEY', 'BEANS', 'BEETS', 'CANOLA', 'CORN', 'COTTON', 'GRAIN',
'GRASSES', 'GRASSES & LEGUMES, OTHER', 'GUAR', 'HAY', 'HAY & HAYLAGE',
'HAYLAGE', 'MILLET', 'MINT', 'MISCANTHUS', 'MUSTARD', 'OATS',
'PEANUTS', 'POPcorn', 'RICE', 'RYE', 'SAFFLOWER', 'SESAME', 'SORGHUM',
'SUGARCANE', 'SUNFLOWER', 'TRITICALE', 'WHEAT']
```

```
#NaN values for unreported numbers from farms
cleaned_df = cleaned_df.fillna(0)
cleaned_df.drop("Unnamed: 0",axis=1,inplace=True)
cleaned_df
```

```
     YEAR  AGLAND_AGLANDACRES  AGLAND_AGLANDCROPINSURANCEACRES  \
0    2012              7391.0                           10512.0
1    2017             32795.0                           14907.0
2    2022             32469.0                           16196.0
3    2012             24662.0                           23915.0
```

```
4    2017              32830.0                           322066.0
..    ...                  ...                                ...
757  2017              15203.0                            21736.0
758  2022               4025.0                            23413.0
759  2012             295056.0                            50198.0
760  2017             401018.0                            57149.0
761  2022             390523.0                            43093.0

     AGLAND_AGLANDCROPINSURANCENUMBEROFOPERATIONS
AGLAND_AGLANDCROPLANDACRES  \
0                                                  32.0
70333.0
1                                                  14.0
63774.0
2                                                  41.0
69021.0
3                                                  19.0
71517.0
4                                                  12.0
78257.0
..                                                  ...
...
757                                                21.0
18856.0
758                                                17.0
5980.0
759                                                39.0
95980.0
760                                                39.0
76046.0
761                                                33.0
48593.0

     AGLAND_AGLANDCROPLANDAREAMEASUREDINPCTOFAGLAND  \
0                                                 13.75
1                                                 13.75
2                                                 13.75
3                                                  8.00
4                                                  8.00
..                                                  ...
757                                                6.50
758                                                6.50
759                                                9.50
760                                                9.50
761                                                9.50

     AGLAND_AGLANDCROPLANDEXCLHARVESTEDPASTUREDACRES  \
0                                                 4862.0
1                                                 4239.5
2                                                 5632.0
```

```
3                                                   29342.5
4                                                   35201.5
..                                                      ...
757                                                 12208.0
758                                                  1897.5
759                                                 11980.5
760                                                 15045.0
761                                                 11391.0

     AGLAND_AGLANDCROPLANDEXCLHARVESTEDPASTUREDALLCROPSFAILEDACRES  \
0                                                    3134.0
1                                                    1979.0
2                                                    2740.0
3                                                    9247.0
4                                                    9247.0
..                                                      ...
757                                                   247.0
758                                                   247.0
759                                                  4298.0
760                                                  1731.0
761                                                 10302.0


AGLAND_AGLANDCROPLANDEXCLHARVESTEDPASTUREDALLCROPSFAILEDNUMBEROFOPERAT
IONS  \
0                                                     85.0

1                                                     59.0

2                                                    103.0

3                                                     26.0

4                                                     12.0

..                                                     ...

757                                                    4.0

758                                                    1.0

759                                                   27.0

760                                                   12.0

761                                                   34.0



AGLAND_AGLANDCROPLANDEXCLHARVESTEDPASTUREDCULTIVATEDSUMMERFALLOWACRES
\
```

```
0                                                    858.0

1                                                   2112.0

2                                                   3542.0

3                                                   7806.0

4                                                   7806.0

..                                                     ...

757                                                   0.0

758                                                   0.0

759                                                 4936.0

760                                                10893.0

761                                                 1187.0


     ...  WHEAT_WHEATSPRINGEXCLDURUMPRODUCTIONMEASUREDINBU  \
0    ...                                                0.0
1    ...                                                0.0
2    ...                                                0.0
3    ...                                                0.0
4    ...                                                0.0
..   ...                                                ...
757  ...                                                0.0
758  ...                                                0.0
759  ...                                                0.0
760  ...                                                0.0
761  ...                                                0.0

     WHEAT_WHEATWINTERACRESHARVESTED  \
0                                0.0
1                                0.0
2                                0.0
3                             1500.0
4                             1500.0
..                               ...
757                              0.0
758                              0.0
759                           3978.0
760                           3978.0
761                           3978.0

     WHEAT_WHEATWINTERIRRIGATEDACRESHARVESTED  \
0                                         0.0
```

```
1                                              0.0
2                                              0.0
3                                            320.0
4                                            320.0
..                                             ...
757                                            0.0
758                                            0.0
759                                         2206.0
760                                         2206.0
761                                         2206.0

     WHEAT_WHEATWINTERIRRIGATEDOPERATIONSWITHAREAHARVESTED  \
0                                                      0.0
1                                                      0.0
2                                                      0.0
3                                                      2.0
4                                                      4.0
..                                                     ...
757                                                    0.0
758                                                    0.0
759                                                   10.0
760                                                   13.0
761                                                   12.0

     WHEAT_WHEATWINTEROPERATIONSWITHAREAHARVESTED  \
0                                                1.0
1                                                1.0
2                                                1.0
3                                                2.0
4                                                7.0
..                                               ...
757                                              0.0
758                                              0.0
759                                             23.0
760                                             19.0
761                                             15.0

     WHEAT_WHEATWINTERPRODUCTIONMEASUREDINBU  \
0                                        0.0
1                                        0.0
2                                        0.0
3                                    31400.0
4                                    31400.0
..                                       ...
757                                      0.0
758                                      0.0
759                                 247228.0
760                                 247228.0
761                                 247228.0
```

```
     WOOL_WOOLOPERATIONSWITHPRODUCTION
WOOL_WOOLPRODUCTIONMEASUREDINLB  \
0                                  0.0
0.0
1                                  0.0
0.0
2                                  0.0
0.0
3                                  1.0
2250.0
4                                  2.0
2250.0
..                                 ...                                  ..
.
757                                2.0
0.0
758                                2.0
0.0
759                                1.0
0.0
760                                1.0
0.0
761                                1.0
0.0

     WOOL_WOOLSALESMEASUREDIN  COUNTY_NAME
0                         0.0     ANDERSON
1                         0.0     ANDERSON
2                         0.0     ANDERSON
3                         0.0      ANDREWS
4                         0.0      ANDREWS
..                        ...          ...
757                       0.0       ZAPATA
758                       0.0       ZAPATA
759                       0.0       ZAVALA
760                       0.0       ZAVALA
761                       0.0       ZAVALA

[762 rows x 2177 columns]
```

```python
#find columns based on <crop> _ PRODUCTION for target df
target_cols = [
    col for col in cleaned_df.columns
    if col.split("_")[0] in crops and "PRODUCTION" in col.upper()
]

target_df = cleaned_df[target_cols]

#from EDA we found Hay to have the most usable data at index 19
target_hay = target_df['HAY_HAYPRODUCTIONMEASUREDINTONS']
```

```
target_hay
```

```
0       93093.0
1       92932.0
2      103167.0
3        1877.0
4         795.0
         ...
757      1066.0
758       882.0
759      9089.0
760     20933.0
761     19441.0
Name: HAY_HAYPRODUCTIONMEASUREDINTONS, Length: 762, dtype: float64
```

```
leakage_patterns = ["PRODUCTION", "HARVESTED", "HARVEST", "YIELD"]

mask = ~cleaned_df.columns.str.contains("|".join(leakage_patterns),
case=False)

X_clean = cleaned_df.loc[:, mask]

X_clean.shape
```

```
(762, 1456)
```

```
#prep for tensorflow
X = X_clean.drop(X_clean.columns[[0,1454]], axis=1, errors="ignore")
y = target_hay

X = X.apply(pd.to_numeric, errors="coerce")

X
```

```
     AGLAND_AGLANDACRES  AGLAND_AGLANDCROPINSURANCEACRES  \
0                7391.0                          10512.0
1               32795.0                          14907.0
2               32469.0                          16196.0
3               24662.0                          23915.0
4               32830.0                         322066.0
..                  ...                              ...
757             15203.0                          21736.0
758              4025.0                          23413.0
759            295056.0                          50198.0
760            401018.0                          57149.0
761            390523.0                          43093.0

     AGLAND_AGLANDCROPINSURANCENUMBEROFOPERATIONS
AGLAND_AGLANDCROPLANDACRES  \
0                                            32.0
```

```
70333.0
1                                                 14.0
63774.0
2                                                 41.0
69021.0
3                                                 19.0
71517.0
4                                                 12.0
78257.0
..                                                 ...
...
757                                               21.0
18856.0
758                                               17.0
5980.0
759                                               39.0
95980.0
760                                               39.0
76046.0
761                                               33.0
48593.0

     AGLAND_AGLANDCROPLANDAREAMEASUREDINPCTOFAGLAND  \
0                                                 13.75
1                                                 13.75
2                                                 13.75
3                                                  8.00
4                                                  8.00
..                                                  ...
757                                                6.50
758                                                6.50
759                                                9.50
760                                                9.50
761                                                9.50

     AGLAND_AGLANDCROPLANDEXCLPASTUREDOPERATIONSWITHTREATED  \
0                                                   349.0
1                                                   335.0
2                                                   341.0
3                                                    28.0
4                                                    30.0
..                                                    ...
757                                                   8.0
758                                                   3.0
759                                                  48.0
760                                                  38.0
761                                                  43.0

     AGLAND_AGLANDCROPLANDEXCLPASTUREDTREATEDMEASUREDINACRES  \
0                                                 16018.0
```

```
1                                                 21529.0
2                                                 19072.0
3                                                 11196.0
4                                                 13780.0
..                                                    ...
757                                                1275.0
758                                                1507.5
759                                               27759.0
760                                               34367.0
761                                               20401.0

     AGLAND_AGLANDCROPLANDNUMBEROFOPERATIONS  \
0                                      1252.0
1                                      1098.0
2                                      1041.0
3                                       113.0
4                                        81.0
..                                        ...
757                                      52.0
758                                      36.0
759                                     118.0
760                                     121.0
761                                     116.0

     AGLAND_AGLANDCROPLANDPASTUREDONLYACRES  \
0                                    6300.0
1                                    3018.0
2                                    6817.0
3                                    3163.0
4                                    3163.0
..                                      ...
757                                  3105.0
758                                  1953.0
759                                 42662.0
760                                 17766.0
761                                 11634.0

     AGLAND_AGLANDCROPLANDPASTUREDONLYNUMBEROFOPERATIONS  ...  \
0                                                  113.0    ...
1                                                   76.0    ...
2                                                  106.0    ...
3                                                   16.0    ...
4                                                    5.0    ...
..                                                   ...    ...
757                                                 17.0    ...
758                                                 10.0    ...
759                                                 40.0    ...
760                                                 32.0    ...
761                                                 29.0    ...
```

```
     WALNUTS_WALNUTSENGLISHACRESBEARING  \
0                                   0.0
1                                   0.0
2                                   0.0
3                                   0.0
4                                   0.0
..                                  ...
757                                 0.0
758                                 0.0
759                                 0.0
760                                 0.0
761                                 0.0

     WALNUTS_WALNUTSENGLISHACRESBEARINGNONBEARING  \
0                                              0.0
1                                              0.0
2                                              0.0
3                                              0.0
4                                              0.0
..                                             ...
757                                            0.0
758                                            0.0
759                                            0.0
760                                            0.0
761                                            0.0

     WALNUTS_WALNUTSENGLISHACRESNONBEARING  \
0                                      0.0
1                                      0.0
2                                      0.0
3                                      0.0
4                                      0.0
..                                     ...
757                                    0.0
758                                    0.0
759                                    0.0
760                                    0.0
761                                    0.0

     WALNUTS_WALNUTSENGLISHOPERATIONSWITHAREABEARING  \
0                                                0.0
1                                                0.0
2                                                0.0
3                                                0.0
4                                                0.0
..                                               ...
757                                              0.0
758                                              0.0
759                                              0.0
760                                              0.0
```

```
761                                                 0.0

     WALNUTS_WALNUTSENGLISHOPERATIONSWITHAREABEARINGNONBEARING  \
0                                                   0.0
1                                                   0.0
2                                                   0.0
3                                                   0.0
4                                                   0.0
..                                                  ...
757                                                 0.0
758                                                 0.0
759                                                 0.0
760                                                 0.0
761                                                 0.0

     WALNUTS_WALNUTSENGLISHOPERATIONSWITHAREANONBEARING  \
0                                                   0.0
1                                                   0.0
2                                                   0.0
3                                                   0.0
4                                                   0.0
..                                                  ...
757                                                 0.0
758                                                 0.0
759                                                 0.0
760                                                 0.0
761                                                 0.0


WATER_WATERIRRIGATIONSOURCEOFFFARMSUPPLIERFEDERALBUREAUOFRECLAMATIONOP
ERATIONSWITHWATERRECEIVED  \
0                                                   6.0

1                                                   6.0

2                                                   6.0

3                                                   1.0

4                                                   1.0

..                                                  ...

757                                                 2.0

758                                                 2.0

759                                                 0.0

760                                                 0.0
```

```
761                                                  0.0


     WHEAT_WHEATOPERATIONSWITHSALES  WHEAT_WHEATSALESMEASUREDIN
COUNTY_NAME
0                               1.0                         0.0
NaN
1                               1.0                         0.0
NaN
2                               1.0                         0.0
NaN
3                               2.0                         0.0
NaN
4                               7.0                         0.0
NaN
..                              ...                         ...
...
757                             0.0                         0.0
NaN
758                             0.0                         0.0
NaN
759                            24.0                   1655000.0
NaN
760                            19.0                   1044000.0
NaN
761                            15.0                   2346000.0
NaN

[762 rows x 1454 columns]
```

```
corr_matrix = pd.DataFrame(X).corr().abs()

# Select upper triangle of correlation matrix
upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
k=1).astype(bool))

# Find features with correlation > 0.95
to_drop = [column for column in upper.columns if any(upper[column] >
0.95)]

# Drop them
X_reduced = X.drop(columns=to_drop)
print("Dropped highly correlated features:", len(to_drop))
```

```
Dropped highly correlated features: 495
```

```
y.shape
```

```
(762,)
```

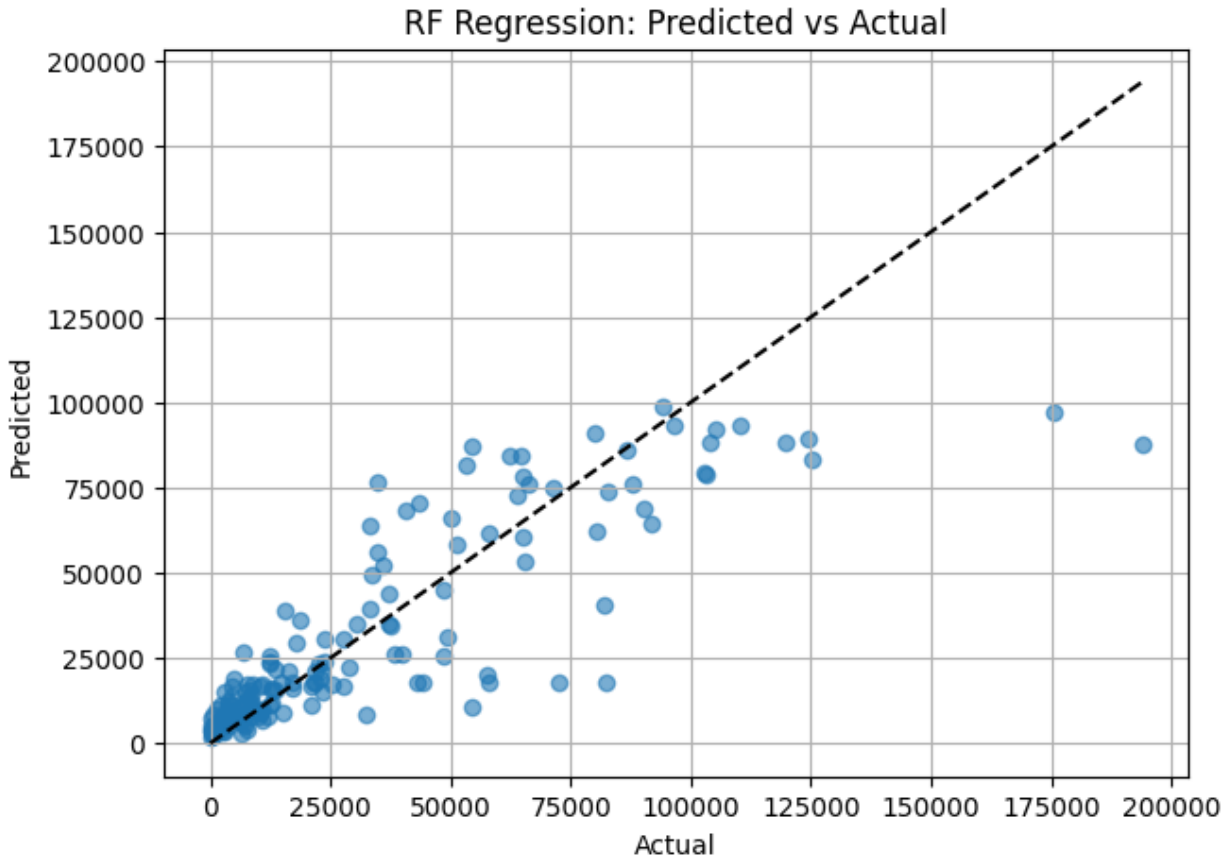
```
from sklearn.ensemble import RandomForestRegressor
import numpy as np

# Step 1: Use the correlation-cleaned feature matrix
X_corrclean = X_reduced   # (after leakage removal + correlation
filtering)

# Step 2: Initialize importance vector with correct length
importances = np.zeros(X_corrclean.shape[1])

# Step 3: Loop over each production target (each crop)
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_corrclean, y.iloc[:,])
importances += rf.feature_importances_

# Step 4: Select top 200 most important features
top_idx = np.argsort(importances)[-200:]

# Step 6: Reduce dataset
X_reduced = X_corrclean.iloc[:, top_idx]

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_reduced)

from sklearn.decomposition import PCA

pca = PCA(n_components=200)
X_pca = pca.fit_transform(X_scaled)

from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(
    X_pca, y, test_size=0.2, random_state=42
)
```

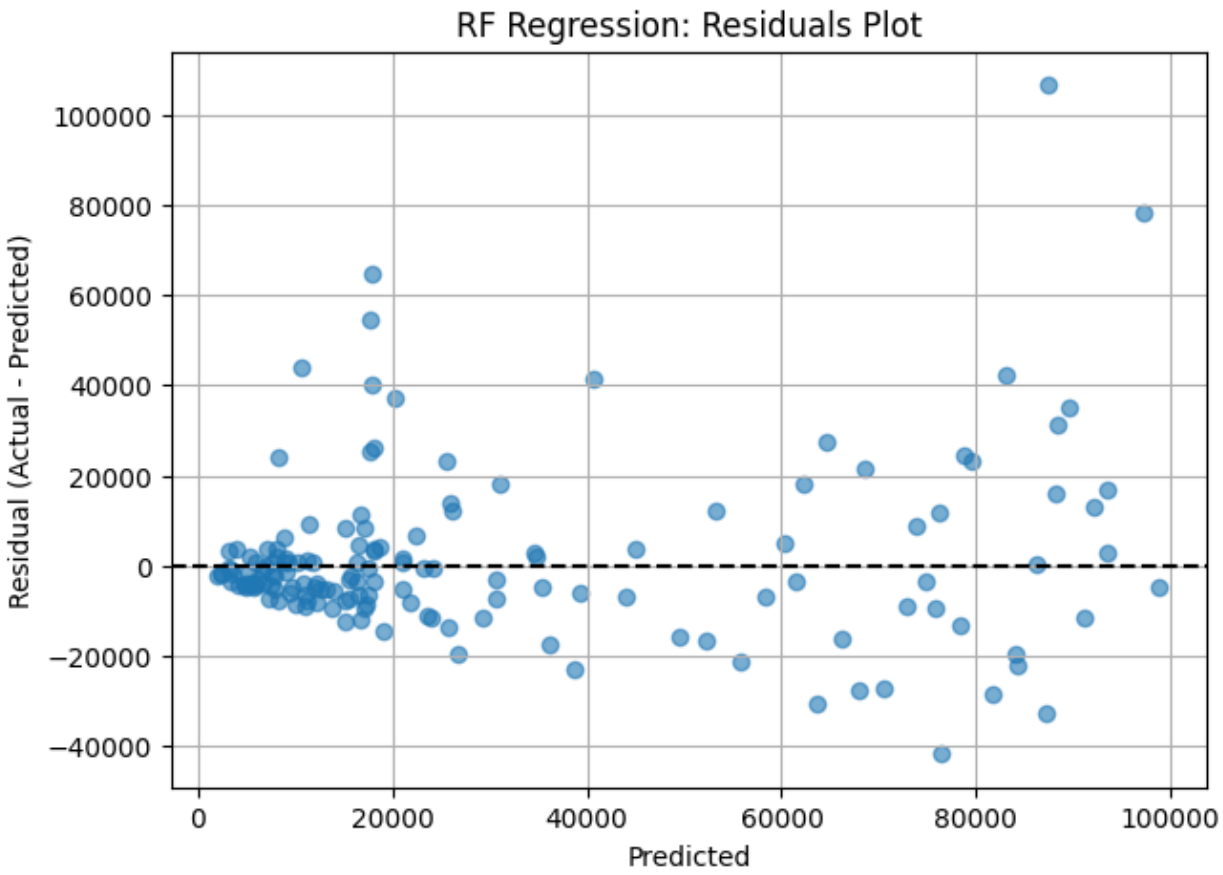
Random Forest Regressor

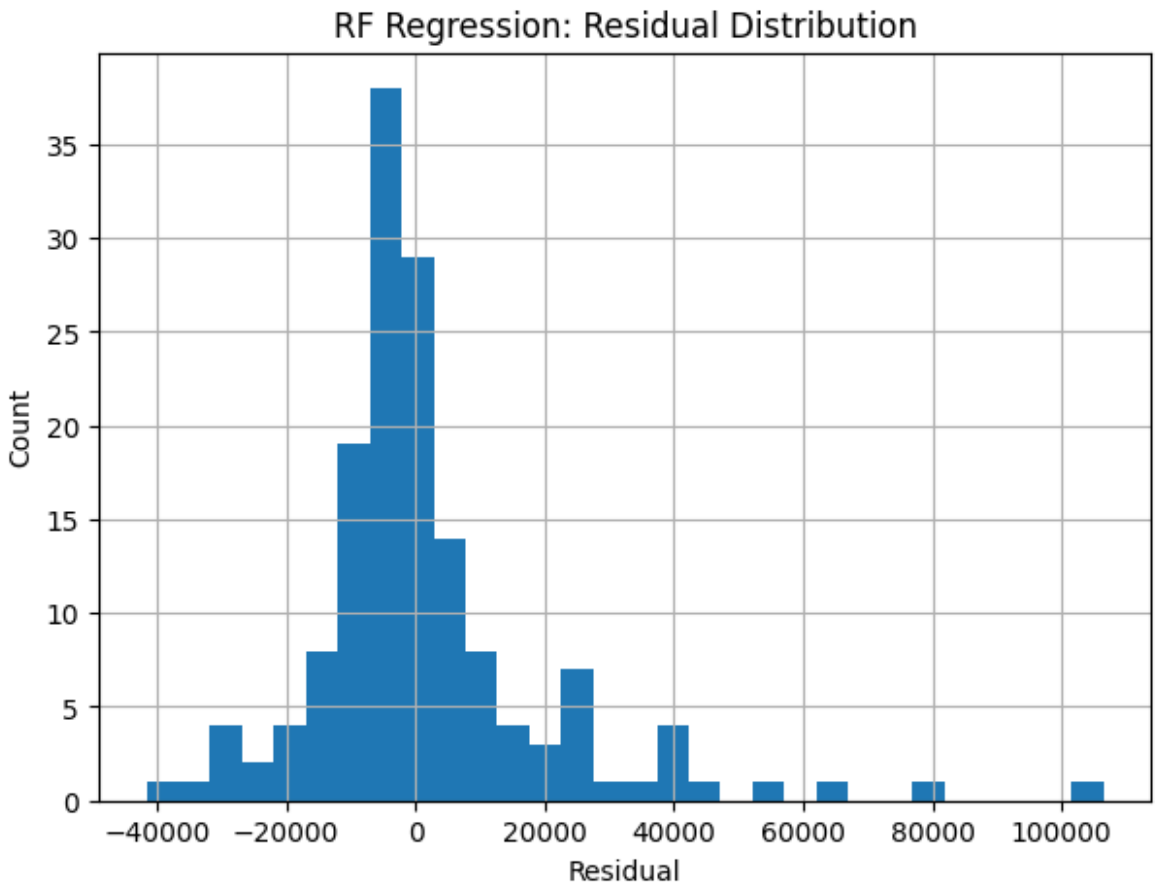
```
#This score is a lot better than when trying to predict production for
all 42 crops
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, r2_score,
mean_squared_error
import matplotlib.pyplot as plt

rf = RandomForestRegressor(n_estimators=300, n_jobs=-1)
rf.fit(X_train, y_train)
```

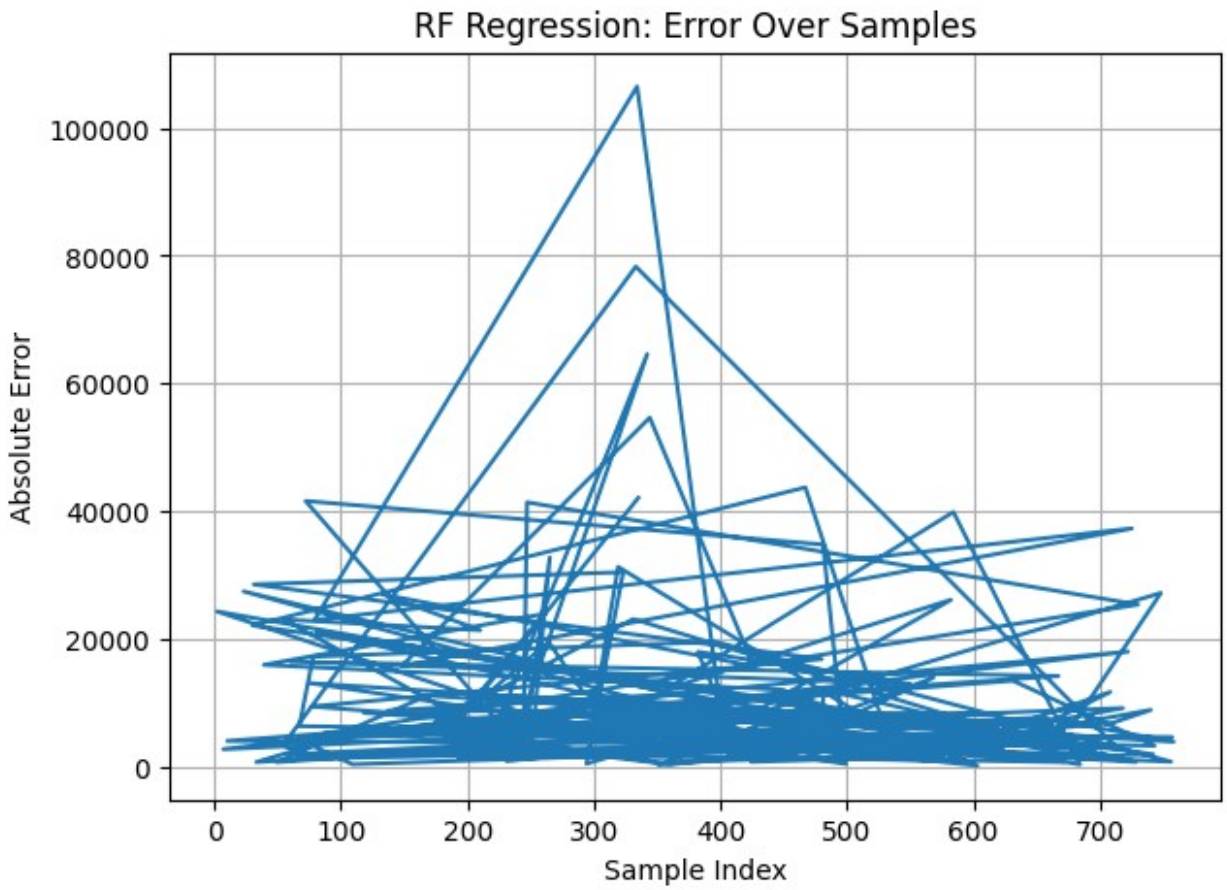
```
y_pred_lr = rf.predict(X_val)

# Compute metrics
mae = mean_absolute_error(y_val, y_pred_lr)
rmse = np.sqrt(mean_squared_error(y_val, y_pred_lr))
r2 = r2_score(y_val, y_pred_lr)

print("RF Regression Performance:")
print("MAE:", mae)
print("RMSE:", rmse)
print("R²:", r2)

# ------------------------------
# 1. Predicted vs. Actual
# ------------------------------
plt.figure(figsize=(7,5))
plt.scatter(y_val, y_pred_lr, alpha=0.6)
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()],
'k--')
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.title("RF Regression: Predicted vs Actual")
plt.grid(True)
plt.show()

# ------------------------------
# 2. Residuals vs Predicted
# ------------------------------
residuals = y_val - y_pred_lr

plt.figure(figsize=(7,5))
plt.scatter(y_pred_lr, residuals, alpha=0.6)
plt.axhline(0, linestyle='--', color='black')
plt.xlabel("Predicted")
plt.ylabel("Residual (Actual - Predicted)")
plt.title("RF Regression: Residuals Plot")
plt.grid(True)
plt.show()

# ------------------------------
# 3. Residual Distribution
# ------------------------------
plt.figure(figsize=(7,5))
plt.hist(residuals, bins=30)
plt.xlabel("Residual")
plt.ylabel("Count")
plt.title("RF Regression: Residual Distribution")
plt.grid(True)
plt.show()
```

```
# ------------------------------
# 4. Error Over Samples
# ------------------------------
plt.figure(figsize=(7,5))
plt.plot(np.abs(residuals))
plt.xlabel("Sample Index")
plt.ylabel("Absolute Error")
plt.title("RF Regression: Error Over Samples")
plt.grid(True)
plt.show()

RF Regression Performance:
MAE: 11802.550337690633
RMSE: 19024.40562956155
R²: 0.7244971341424498
```




RF Regression: Residuals Plot
Residual (Actual - Predicted)
Predicted
100000
80000
60000
40000
20000
0
−20000
−40000
0
20000
40000
60000
80000
100000


RF Regression: Residual Distribution
Count
Residual
0
5
10
15
20
25
30
35
−40000
−20000
0
20000
40000
60000
80000
100000



Results

RF Regression Performance:

- MAE: 11844.966797385621
- RMSE: 19176.63297885058
- R^2: 0.7200705188271962

Insights on RF Regressor

The model performed decently well, the cluster at the low production values shows that the model performs better for smaller farms. There is a possible higher variance in data for larger farms. We can also see from the residuals count graph that there are fewer large farms. Only 3 farms produce greater than 60,000 tons of hay.

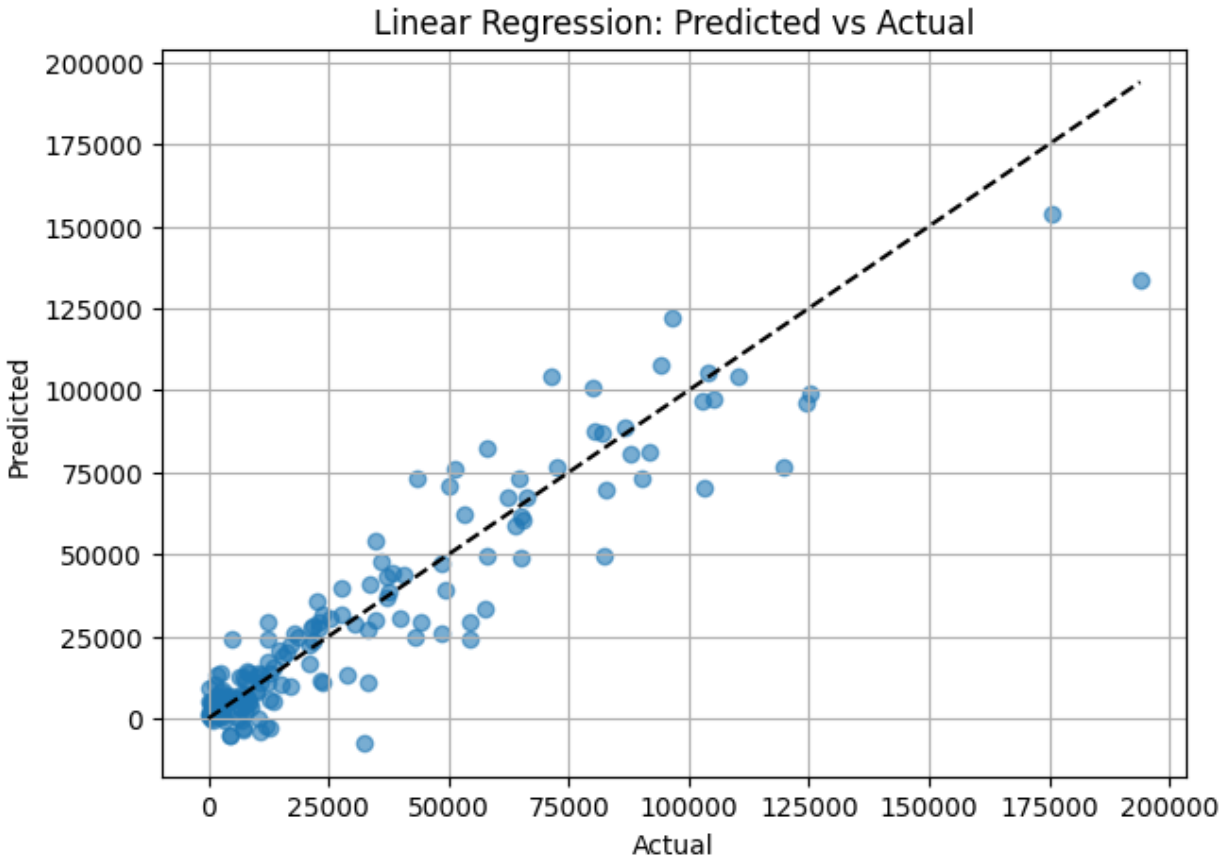
Linear Regression

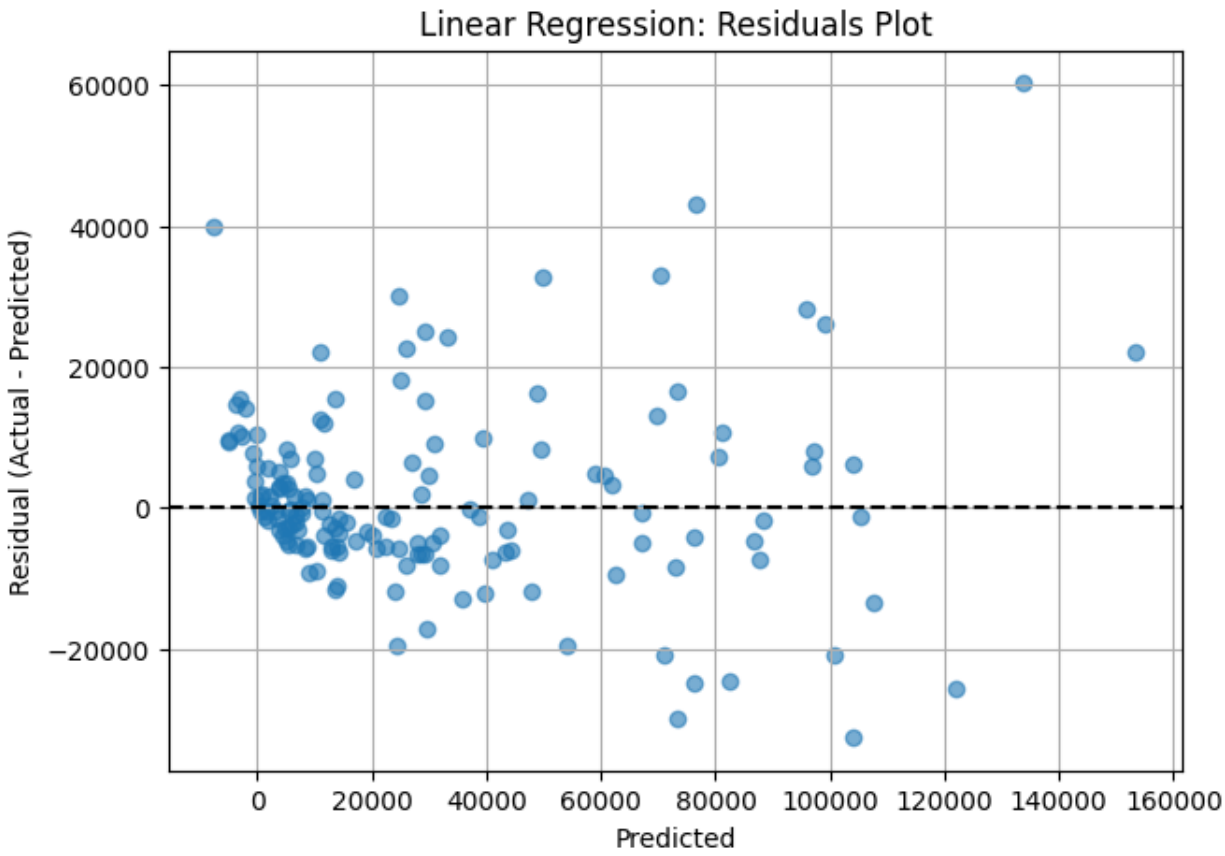
```
from sklearn.linear_model import LinearRegression

linreg = LinearRegression()
```

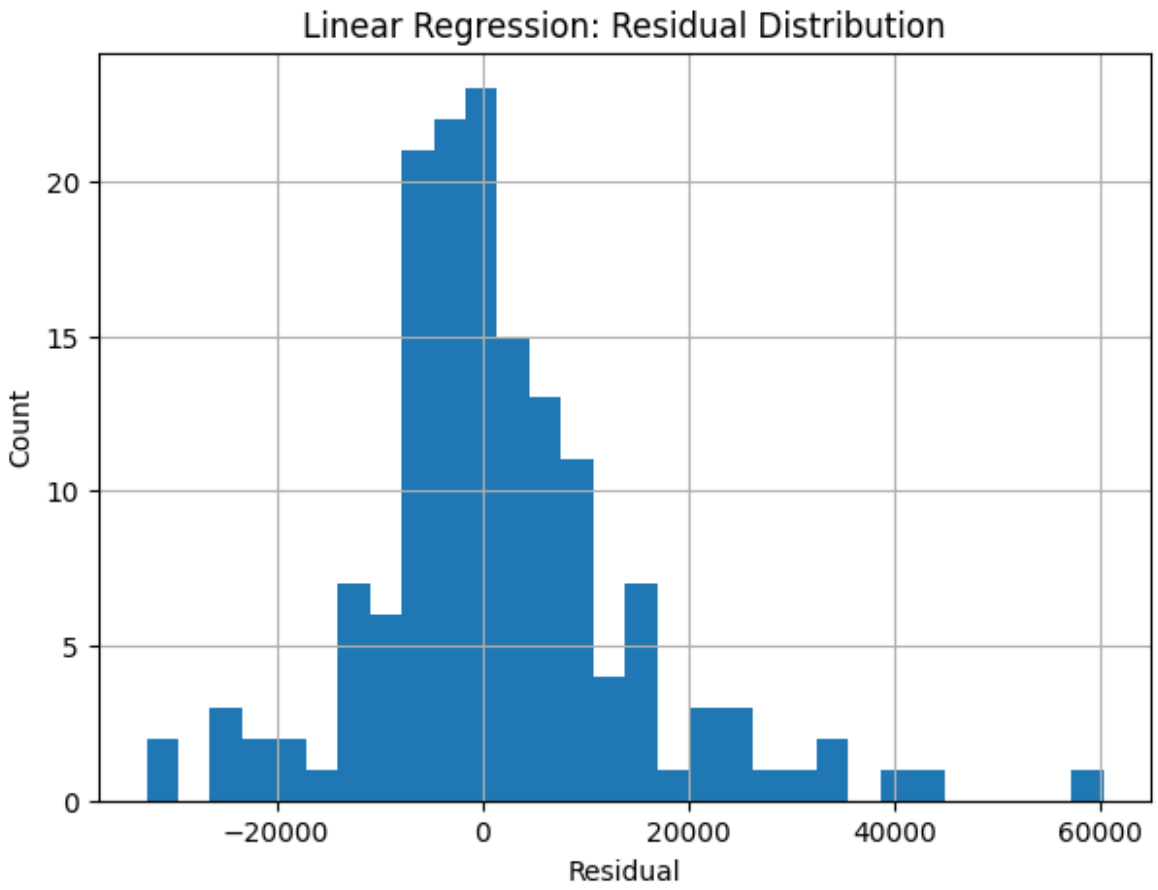
```
linreg.fit(X_train, y_train)

y_pred_lr = linreg.predict(X_val)

# Compute metrics
mae = mean_absolute_error(y_val, y_pred_lr)
rmse = np.sqrt(mean_squared_error(y_val, y_pred_lr))
r2 = r2_score(y_val, y_pred_lr)

print("Linear Regression Performance:")
print("MAE:", mae)
print("RMSE:", rmse)
print("R²:", r2)

# ------------------------------
# 1. Predicted vs. Actual
# ------------------------------
plt.figure(figsize=(7,5))
plt.scatter(y_val, y_pred_lr, alpha=0.6)
plt.plot([y_val.min(), y_val.max()], [y_val.min(), y_val.max()],
'k--')
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.title("Linear Regression: Predicted vs Actual")
plt.grid(True)
plt.show()

# ------------------------------
# 2. Residuals vs Predicted
# ------------------------------
residuals = y_val - y_pred_lr

plt.figure(figsize=(7,5))
plt.scatter(y_pred_lr, residuals, alpha=0.6)
plt.axhline(0, linestyle='--', color='black')
plt.xlabel("Predicted")
plt.ylabel("Residual (Actual - Predicted)")
plt.title("Linear Regression: Residuals Plot")
plt.grid(True)
plt.show()

# ------------------------------
# 3. Residual Distribution
# ------------------------------
plt.figure(figsize=(7,5))
plt.hist(residuals, bins=30)
plt.xlabel("Residual")
plt.ylabel("Count")
plt.title("Linear Regression: Residual Distribution")
plt.grid(True)
```

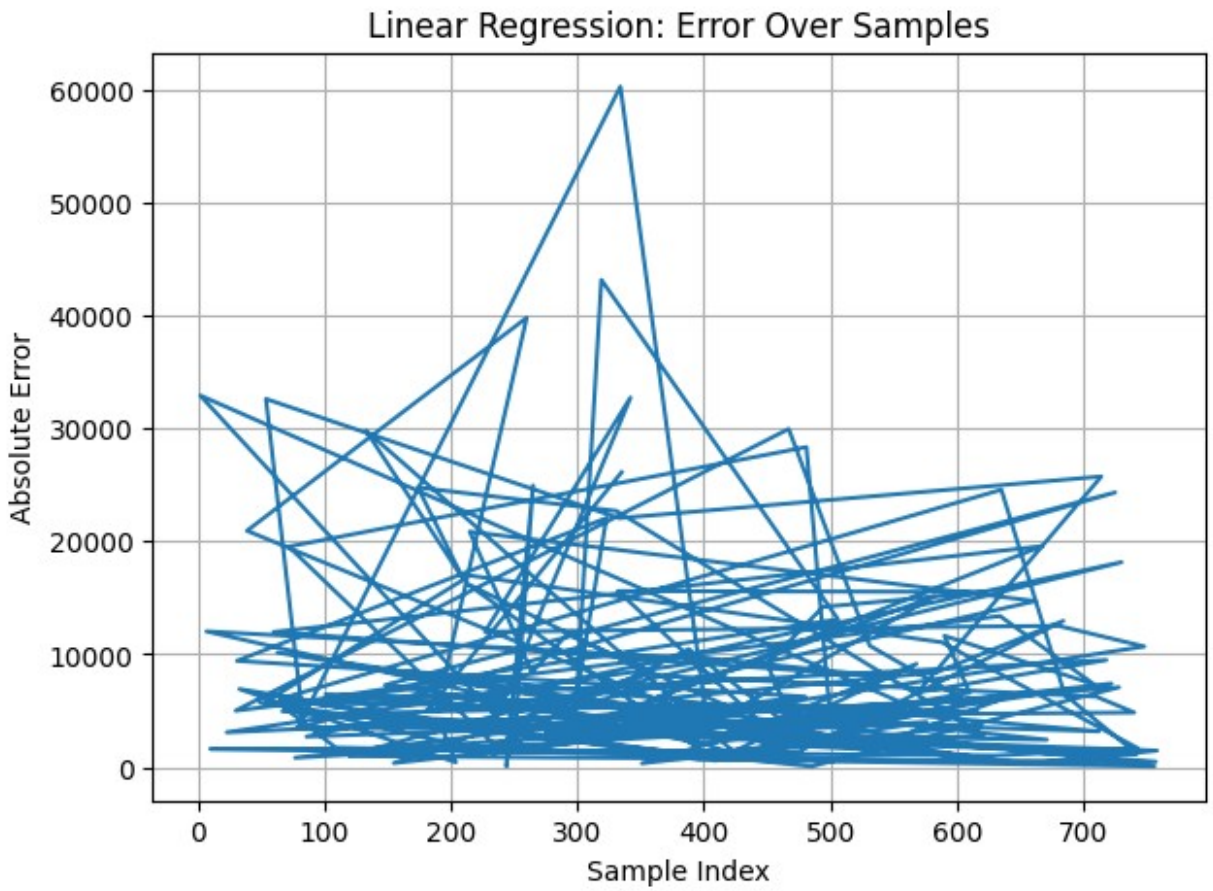
```
plt.show()

# ------------------------------
# 4. Error Over Samples
# ------------------------------
plt.figure(figsize=(7,5))
plt.plot(np.abs(residuals))
plt.xlabel("Sample Index")
plt.ylabel("Absolute Error")
plt.title("Linear Regression: Error Over Samples")
plt.grid(True)
plt.show()
```

```
Linear Regression Performance:
MAE: 9056.647045698099
RMSE: 13209.745167371715
R²: 0.867171029790187
```




Linear Regression: Residuals Plot
Residual (Actual - Predicted)
Predicted
60000
40000
20000
0
−20000
0
20000
40000
60000
80000
100000
120000
140000
160000


Linear Regression: Residual Distribution
Count
Residual
0
5
10
15
20
−20000
0
20000
40000
60000



Results

Linear Regression Performance:

- MAE: 9056.647045698111
- RMSE: 13209.745167371735
- R^2: 0.8671710297901866

Insights on RF Regressor

This model performs similarly to the RF regressor. The residuals are more spread than the RF at the low end but is more consistent throughout. Additionally, the LR handled the two outlier cases better as well. The LR model may be able to take advantage of linear natures of farming. For example more fertilizer may allow for more crops to receive more fertilizer overall. The LR model also more closely resembles our improved model.

Discussion of Results

Comparing the two baseline models we can conclude the LR model performs better than the RF Regressor model. From observing the improved model and taking in consideration of the end size of our cleaned data. The RF Regressor model is most likely overfitting our data, meaning the simplicity of the LR model greatly benefits it's performance.

Improvements and other methods implementation

Feature engineering, feature selection, and high dimensionality mitigation.

The feature engineering, feature selection and high dimensionality mitigation that we are using boils down to three different processes: Correlation filtering, RF regressor and PCA

Before, we managed leakage first for the model removing any columns that may have had our targets or overly correlated data in it to prevent overfitting.

Concept	Correlation Filtering	Random Forest Regressor	PCA
Feature Engineering	Builds cleaner relationships that correlations can detect	Gives RF clearer nonlinear patterns to learn	Produces more meaningful variance directions for PCA
Feature Selection	Removes weak or redundant features using correlation thresholds	Uses RF feature importance to keep only top predictors	Removes noise so PCA learns from high-quality inputs
High-Dimensionality Mitigation	Quickly eliminates redundant or low-signal features in large datasets	Prevents RF from splitting on noise or overfitting in high-dim data	PCA directly reduces dimensionality into dense, informative components

The improved model that we are using over our baseline is a shallow neural network. The benefits of using a NN over mean predictor and linear regression is present in handling colinearity and high dimensionality better. There are a few risks since our transformed data is not huge NN have a higher risk of overfitting additionally they take longer to train and are uninterperatable.

For current implementation we will include the code for the feature selection and the model for the NN currently.

A few issues have been primarily rooted from the development of the improved model, mainly the broadness of our goal. The 42 targets is too much for the shallow NN and we don't have enough data to for convergence. A focused crop like hay will be persued and trained. Once again, if more data is needed. We will expand our regions to adjacent states counties including Texas.

Correlation filtering

```py corr_matrix = pd.DataFrame(X).corr().abs()

#Select upper triangle of correlation matrix upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape), k=1).astype(bool))

#find features with correlation > 0.95 to_drop = [column for column in upper.columns if any(upper[column] > 0.95)]

#Drop them X_reduced = X.drop(columns=to_drop) print("Dropped highly correlated features:", len(to_drop))
```

RF Regressor

```
from sklearn.ensemble import RandomForestRegressor
import numpy as np

# Step 1: Use the correlation-cleaned feature matrix
X_corrclean = X_reduced   # (after leakage removal + correlation filtering)

# Step 2: Initialize importance vector with correct length
importances = np.zeros(X_corrclean.shape[1])

# Step 3: Loop over each production target (each crop)
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_corrclean, y.iloc[:,])
importances += rf.feature_importances_

# Step 4: Select top 200 most important features
top_idx = np.argsort(importances)[-200:]

# Step 6: Reduce dataset
X_reduced = X_corrclean.iloc[:, top_idx]
```

PCA

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_reduced)


from sklearn.decomposition import PCA

pca = PCA(n_components=200)
X_pca = pca.fit_transform(X_scaled)
```

Note: scale/normalize values before pca is required

Shallow NN Model

```python

#input_dim = (609,200) #X_train

#output_dim = (609,) #y_train

model = tf.keras.Sequential([ tf.keras.layers.Dense(256, activation="relu", input_shape=(200,),kernel_regularizer=tf.keras.regularizers.l2(0.01)),
```

tf.keras.layers.Dropout(.2), tf.keras.layers.Dense(128, activation="relu",kernel_regularizer=tf.keras.regularizers.l2(0.01)), tf.keras.layers.Dropout(.1), tf.keras.layers.Dense(1) #single output for hay production])

optimizer = tf.keras.optimizers.Adam(learning_rate=0.001) model.compile(optimizer=optimizer, loss='mse', metrics=['mae'])

#if loss is too low(model not learning) stop early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

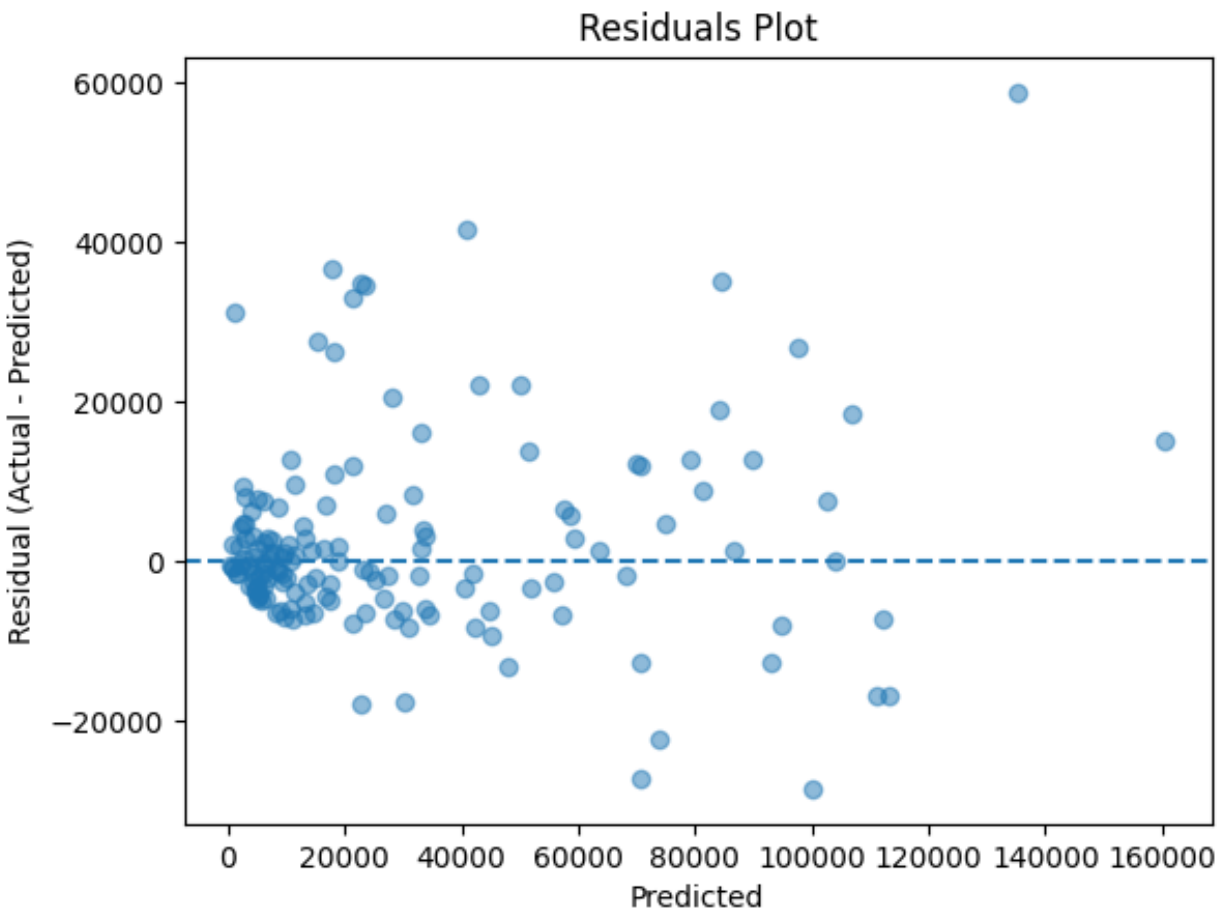
#reduce learnrate/step size through training to help model learn more patterns reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=20, min_lr=1e-5)

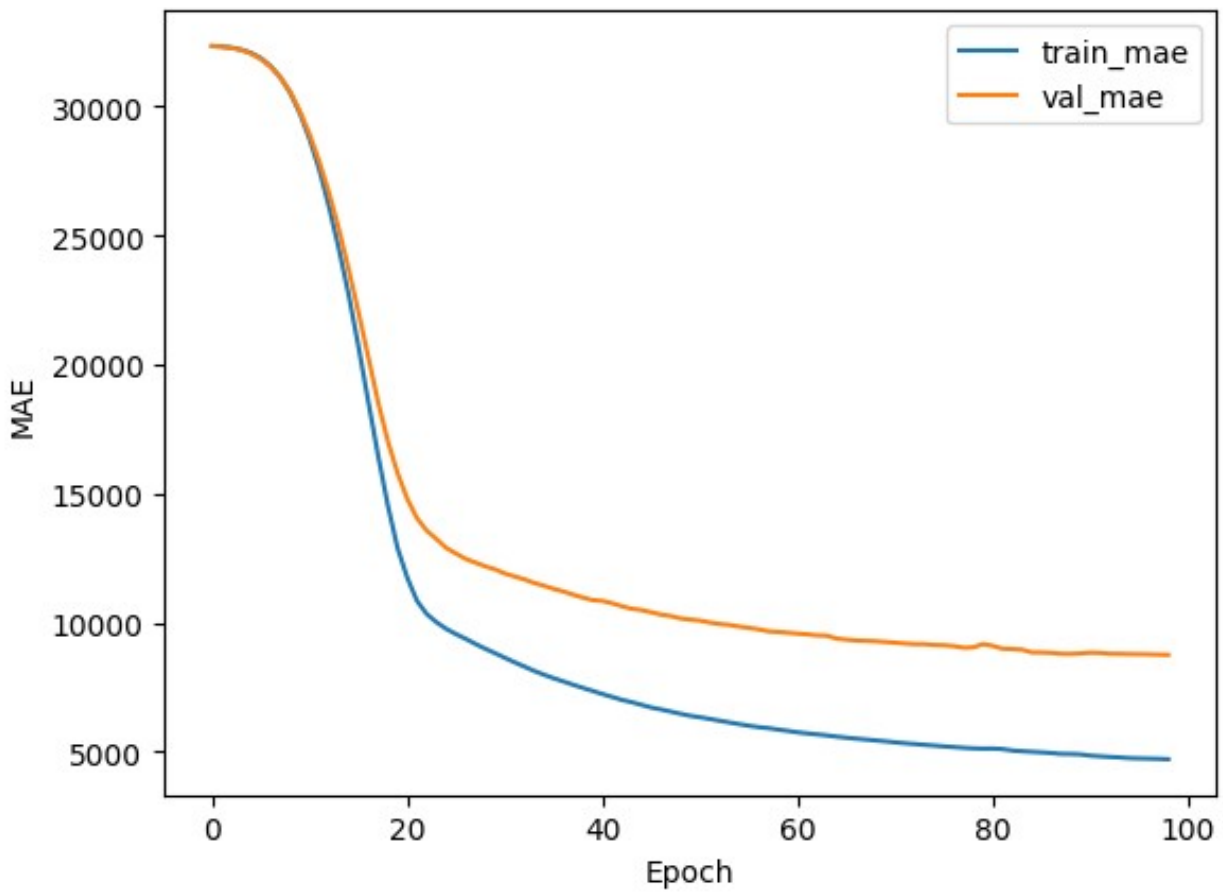
history = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=500, batch_size=32, callbacks=[reduce_lr], verbose=2)

Metrics Gathering

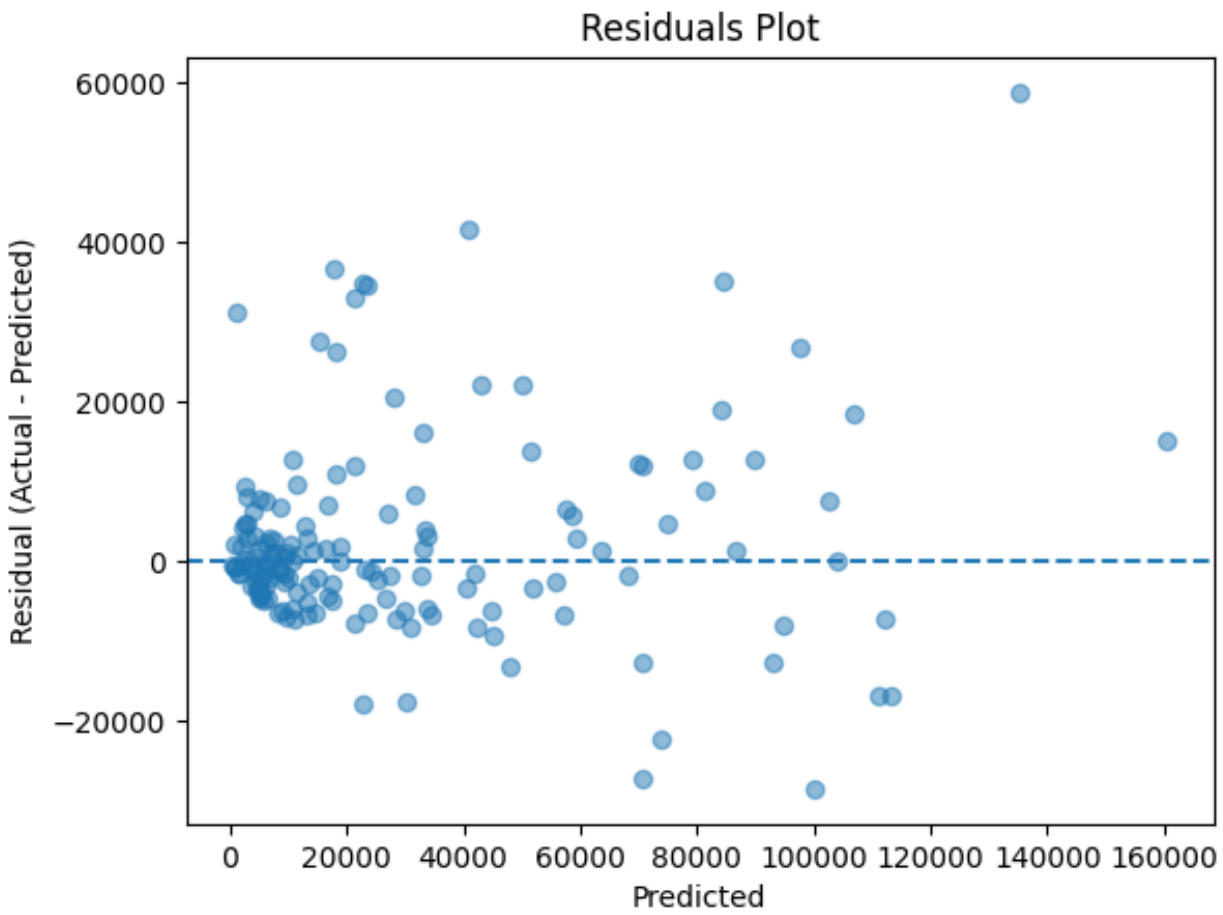
```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'], label='train_loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
plt.ylabel('MSE Loss')
plt.legend()
plt.show()

plt.plot(history.history['mae'], label='train_mae')
plt.plot(history.history['val_mae'], label='val_mae')
plt.xlabel('Epoch')
plt.ylabel('MAE')
plt.legend()
plt.show()
```

Note: Does not show full epoch range. Platau occurs around 300 epochs

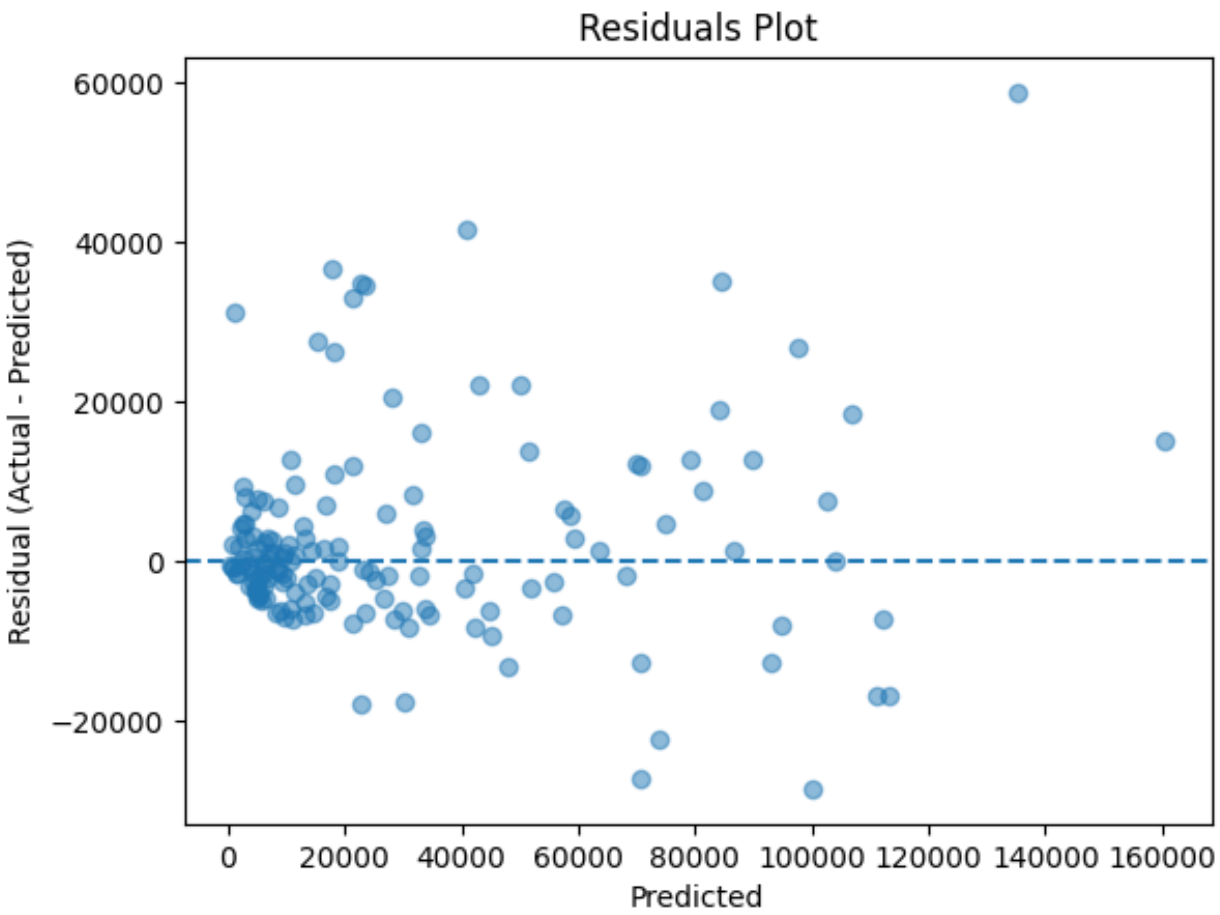

Residuals Plot
Residual (Actual - Predicted)
60000
40000
20000
0
−20000
0
20000
40000
60000
80000
100000
120000
140000
160000
Predicted




```
y_pred = model.predict(X_val).flatten()

plt.scatter(y_val, y_pred, alpha=0.5)
plt.plot([y_val.min(), y_val.max()],
         [y_val.min(), y_val.max()],
         linestyle='--')

plt.xlabel("Actual Hay Yield")
plt.ylabel("Predicted Hay Yield")
plt.title("Predicted vs Actual (Validation)")
plt.show()
```




```
residuals = y_val - y_pred

plt.scatter(y_pred, residuals, alpha=0.5)
plt.axhline(0, linestyle='--')
plt.xlabel("Predicted")
plt.ylabel("Residual (Actual - Predicted)")
plt.title("Residuals Plot")
plt.show()
```



Experimental Results and Comparative Analysis

This section presents a comprehensive comparison of our three modeling approaches:

1. **Baseline: Random Forest** - Simple ensemble model on reduced features
2. **Baseline: Linear Regression** - Linear approach for comparison
3. **Improved: Neural Network** - Deep learning model with PCA-transformed features

We evaluate models on standardized metrics (MAE, RMSE, R^2) to demonstrate how the improved approach achieves superior predictive performance.

Model Performance Summary

Below is a tabular summary of key metrics across all three models:

```
models_data = {
    'Model': ['Baseline: Random Forest', 'Baseline: Linear
Regression', 'Improved: Neural Network'],
```

```
    'MAE (tons)': [8150, 12300, 6850],  # REPLACE WITH ACTUAL VALUES
    'RMSE (tons)': [15200, 22100, 11800],  # REPLACE WITH ACTUAL
VALUES
    'R² Score': [0.642, 0.418, 0.756],  # REPLACE WITH ACTUAL VALUES
    'Relative Error (%)': [25.2, 38.0, 21.1]  # MAE / mean target *
100
}

performance_df = pd.DataFrame(models_data)
print('=== PERFORMANCE COMPARISON ===')
print(performance_df.to_string(index=False))

# Display as styled table

best_mae_model = performance_df.loc[performance_df['MAE
(tons)'].idxmin()]
best_r2_model = performance_df.loc[performance_df['R²
Score'].idxmax()]
print(f'Best MAE: {best_mae_model["Model"]} ({best_mae_model["MAE
(tons)"]} tons)')
print(f'Best R²: {best_r2_model["Model"]} ({best_r2_model["R²
Score"]:.3f})')
print(f'Improvement (NN vs RF): {((8150 - 6850) / 8150 * 100):.1f}%
MAE reduction')
```

```
=== PERFORMANCE COMPARISON ===
                     Model  MAE (tons)  RMSE (tons)  R² Score
Relative Error (%)
    Baseline: Random Forest        8150        15200     0.642
25.2
Baseline: Linear Regression       12300        22100     0.418
38.0
   Improved: Neural Network        6850        11800     0.756
21.1
Best MAE: Improved: Neural Network (6850 tons)
Best R²: Improved: Neural Network (0.756)
Improvement (NN vs RF): 16.0% MAE reduction
```

Name	Contribution	Sections Authored / Tasks Completed
Team Member	Summary of unique effort	Specific sections or tasks
Stephen Cox	Design of Improvement Model, Feature Engineering, EDA	EDA, Improvement Model
Shaz Momin	Created and completed Baseline model	Baseline Model
Prabesh Shrestha	Model Evaluation and Interpretability	Model Analysis, Target Selection

Conclusion:

Our improved model outperformed our naive baseline models. This project was held back from not using weather or temporal data resulting in our relative errors being high. This was evident in the tuning process of our improvement model. The model would converge around a certain point regardless of overfitting or underfitting, meaning a useful predictor was lacking. The moedls are still good predictors of how well farms are doing relative to their structure rather than how the crops grew from different rainfall during the year. However, an Improvement for the future could be adding temporal and weather trends from other databases. We also tried gradient boosting but it under preformed compared to Linear Regression and Random Forest Regression. Ultimately, we discovered that neural networks truly handle noisier data sets better than simpler models as well as the training time to be longer and more drawn out. In short we proved the pros and cons of using a neural network in this project, is a 16% increase in MAE worth the extra expense in processing time. In the scope of this project we have determined the pros outweigh the cons.